EVALUATING ENERGY-AWARE TASK ALLOCATION STRATEGIES FOR MPSOCS

Fabio Wronski, Eduardo Wenzel Brião, Flávio Rech Wagner
Universidade Federal do Rio Grande do Sul
Instituto de Informática

Abstract: Because of current market trends, the evaluation of task allocation strategies in multiprocessor system-on-chips (MPSoCs) must take into account both performance and energy consumption. Furthermore, complex interconnection structures, such as networks-on-chip (NoCs), must be considered. Simulators for the evaluation of energy consumption of detailed communication patterns in NoCs are available, as well as performance simulators that consider detailed task execution in processors. However, in order to evaluate task allocation strategies in MPSoCs, these two types of simulation models must be combined, since communication and computation events interfere with each other. Besides that, this simulator must implement low-power mechanisms, such as dynamic voltage scaling (DVS), in order to evaluate allocation algorithms that explore the trade-off between performance and energy. A cycle-accurate simulation of the processor and communication behaviors, however, would be too time-consuming, making impossible a fast exploration of different allocation algorithms. This work presents an MPSoC simulator that implements the appropriate abstractions for a precise evaluation of the energy consumption of task allocation algorithms that explore DVS, which is based on the scheduling of synthetic task graphs. A NoC mesh topology is considered, due to its simplicity and scalability. Experiments that implement the allocation of task graphs using different bin-packing heuristics combined with DVS demonstrate the energy-performance design space that may be explored by task allocation algorithms.

Keywords: Networks-on-Chip, Task Allocation, Energy Estimation, Simulation, Multiprocessor SoCs.

1. INTRODUCTION

Due to technology and market trends, multiprocessor systems-on-chip (MPSoCs) are being increasingly used as platforms for embedded systems. These systems require a highly scalable and parallel communication infrastructure, such as networks-on-chip (NoCs). Especially mesh networks have been proposed, due to their simplicity. Energy consumption is one of the critical aspects in the design of embedded systems. Low-power techniques, such as voltage scaling[1], are essential and must also be applied in the context of NoC-based MPSoCs.

MPSoCs run several parallel tasks, which must be allocated into the various processors. Allocation algorithms have been usually evaluated in terms of performance and timing metrics, but now energy consumption also becomes a very relevant factor.

In this context, several estimation tools have been proposed. Examples are SimDVS[2], for evaluating dynamic voltage scaling (DVS); CAFES[3], for evaluation of communication costs; and Xpipes[4], which is based on the Orion[5] library, for energy estimation. SimDVS simulates only individual processors, while Xpipes and CAFES simulate only the NoC infrastructure.

For the evaluation of task allocation strategies, however, both the task scheduling and execution in the processors and the communication traffic must be simulated together, since they interfere which each other. A cycle-accurate simulation of the processor and communication behaviors, however, would be too time-consuming, making impossible a fast exploration of different allocation algorithms.

This work presents an MPSoC simulator that implements the appropriate abstractions for a precise evaluation of the energy consumption of task allocation algorithms that explore DVS, which is based on the scheduling of synthetic task graphs. The simulator has been implemented in SystemC TLM (transaction level model), which offers the required abstractions and synchronization mechanisms.

The processor model allows the application of DVS. For the calculation of the energy consumed by each synthetic task running in the processor, a random parameter representing the number of switching gates per cycle has been included in the model. For a more precise evaluation, task scheduling costs are also considered. The energy consumed by the NoC in the communication between tasks allocated to different processors is evaluated by the Orion library, as in Xpipes.

In the experiments, synthetic task graphs have been allocated using simple bin-packing heuristics. Results show the benefits of DVS, even when inefficient allocations are applied and generate too much communication through the network. The trade-offs between energy and performance may

be evaluated and suggest that reducing the communication is a good heuristic for minimizing the energy consumption, although the combined use of DVS brings more significant gains.

The remaining of this paper is organized as follows. Section 2 discusses related work. The energy and task models are presented in Section 3. The simulator implementation is discussed in Section 4. Section 5 shows experimental results, and Section 6 draws main conclusions and discusses future work.

2. RELATED WORK

As the dynamic power consumption is dominating in CMOS circuits, several architecture-level simulators have been developed to evaluate and estimate power consumption caused by the charging and discharging of the capacitive load on each gate output of the circuit. Two techniques in the literature are often used: calculation of power consumption by the average number of gates switching in processor instructions[6], and the calculation of power consumption by the number of gates switching in architectural components[7].

A simulator for NoC architectures is Xpipes[4]. It was developed in SystemC and consists of a customizable library of network components, which can be configured and instantiated by the developer. The calculation of energy is achieved using the Orion[5] library. This library has low-level capacitance models of NoC router components that allow a more accurate estimation of dynamic power consumption. These components can be buffers, crossbars, and arbiters.

SimDVS[2] is a unified simulation environment for evaluating dynamic voltage scaling (DVS) algorithms, which has task graphs as input. Some models of state-of-the-art processors are already embedded in the environment, but other models can be developed and included by the users. However, SimDVS supports only single processors, not including models for communication architectures like NoCs.

There are several heuristic approaches for task allocation in NoCs that try to minimize energy consumption or optimize other metrics, under energy constraints. Hu and Marculescu[8], for instance, present an approach for applying a static task allocation strategy in NoC architectures under real-time constraints, aiming at energy reduction. The application is modeled by a graph based on a Communication Weight Model (CWM), where arc weights represent the amount of bits exchanged between vertexes. Their approach, however, does not consider the exact simulation of tasks running in the

processors. Besides, our work aims at the evaluation of different allocation approaches and not at the implementation of a single approach.

The CAFES[3] simulator is a framework aiming at the evaluation of several communication models that may be used for the analysis of energy consumption in NoC architectures. However, CAFES does not consider scheduling costs.

In this work, scheduling costs are considered. Besides, our simulator uses the same Orion library for power estimation as Xpipes, but we consider only the mesh architecture with some configurable parameters, instead of a broader library. We rely on a processor model that allows application scheduling, using a task graph model approach similar to SimDVS, while Xpipes uses cycle accurate models for simulating the cores. This Xpipes approach makes high-level energy estimation from task graphs unfeasible.

3. ENERGY AND TASK MODELS

Our energy model considers only dynamic energy consumption, which is still dominant in current technologies, although static power is becoming important. A static energy model is being currently implemented.

In a same time interval, two different task allocations generate different gate switching patterns in the processors and thus different dynamic energy consumptions. In CMOS circuits, the dynamic energy consumption is expressed as:

$$E = \frac{\alpha C}{2} V_{dd}^2$$

where C is a constant that represents the gate capacitance in a given technology; α is a number of gate switchings; and V_{dd} is the circuit supply voltage. If voltage scaling is applied, this parameter is not fixed.

In a processor, each instruction generates a different value for α, which depends on the previous processor state. In a system composed by n tasks $k \in K$, the total α is expressed as:

$$\alpha_{total} = \sum_{i=1}^{n} k_i^{\alpha_{total}}$$

where $k_i^{\alpha_{total}}$ is the total number of switchings of task k_i.

Variable V_{dd} presents a quadratic factor, and it is thus natural that the minimization of supply voltage leads to important energy savings. But reducing the voltage causes the frequency operation to be reduced too, following the equation below[9]:

$$V_{norm} = \beta_1 + \beta_2 f_{norm}$$

where $\beta_1 = V_{th} / V_{max}$ and V_{th} is the threshold voltage, $\beta_2 = 1 - \beta_1$, and V_{norm} and f_{norm} are the voltage and the frequency, both normalized regarding V_{max} and f_{max}. For a 100 nm technology, we have $\beta_1 = 0.3$.

A metric for power consumption evaluation in NoCs is Bit Energy[10]. It defines the power consumed when a data bit is transferred between two routers. This way, we define the energy spent by the transfer of a single bit from node p_i to node p_j as:

$$E_{bit(i,j)} = (\eta - 1) \times (E_{S_{bit}} + E_{B_{bit}} + E_{L_{bit}})$$

where η is the number of routers in the way from p_i to p_j, $E_{S_{bit}}$ is the energy spent in the crossbar inside one router, $E_{L_{bit}}$ is the energy spent in communication links, and $E_{B_{bit}}$ is the energy spent in the buffers. We don't consider the energy $E_{L_{bit}}$ spent in the links, since energy consumption in the routers is much larger.

The Orion[5] library implements an energy estimator for the crossbar and the buffers inside the router. Buffers are usually responsible for 90% of the energy consumption in the router.

The task and communication architecture models used in this work and presented below are based on Hu and Marculescu[8].

Each application is a directed acyclic graph $T = G(K, A)$, where each node $k_i \in K$ is a periodic task and each arc $a_{i,j} \in A$ is a dependency or flow of messages between tasks k_i and k_j. The arc weight $a_{i,j}^W$ represents the amount of bits to be transferred between the tasks.

Each task $k_i \in K$ is a tuple $\{C, T, D, \alpha\}$, where C is the worst case execution, T is the task period, D is the task deadline, and α is the average number of gate switchings per cycle of the task in the core.

The communication architecture is represented by a directed graph $G(P, R)$, where each node $p_i \in P$ represents a core and each directed arc $r_{i,j} \in R$ is a path between p_i and p_j. A core $p_i \in P$ has its own operation frequency p_i^F, cycle period p_i^T and voltage supply p_i^V. In our work, the communication architecture graph represents a mesh network.

4. SIMULATOR

The simulator has been implemented in SystemC, due to its support to the development and simulation of hardware models with the appropriate abstractions for our purposes.

The model of the routers has been implemented at the RT level and is thus very precise in terms of timing and energy. The core model, in turn, in fact considers only the task scheduler, which simulates the states through which the processor goes during the execution of the synthetic tasks. This simplification makes possible a high-level and fast evaluation of the energy

consumption, without requiring the development and execution of real applications. As a drawback, accuracy is lost, since the model has a statistical nature.

The simulator uses the RaSoC[11] routers, designed for the synthesis of low power and low area NoC-based embedded systems in FPGAs. The RaSoC architecture utilizes wormhole packet switching, XY routing algorithm, and handshake control flow. Its energy evaluation was implemented with the Orion library. Each router has 5 bi-directional ports with input buffer size of 4 phits. The phit size is 4 bytes.

The task scheduler in each core is based on the implementation of a priority-based scheduling kernel. The scheduling policy implements an EDF (earliest deadline first) algorithm. For implementing the task dependencies, a list-scheduling algorithm was combined with EDF. Additionally, the AVR[12] (average rate) algorithm was implemented for DVS.

Each task is characterized with respective WCETs and gate switchings per cycles. With these data, tasks are scheduled and their energy consumptions are evaluated. Since the worst case execution time rarely occurs when the task is executed, there is a slack in the scheduling. To simulate a more realistic behaviour, a parameter called *slack* is used to define a range between the best and worst case execution times. By default, it has a value of 30%, which means that the minimum value of the best case execution time may represent 70% of the WCET. For each task execution, the simulator randomly chooses a value for its current execution time, within the allowable range. In addition, the scheduler time and costs are also considered, including the cost of an "idle task", which runs when no other task is available.

5. EXPERIMENTS

For the experiments, synthetic task graphs generated by the TGFF tool[13] has been used. The scheduler costs are obtained by simulation, with the CACO-PS[7] tool, of an API RTSJ[14] (Real-Time Specification for Java) for the FemtoJava[15] processor. The technology selected for the experiments is 100 nm. The NoC is a 3 x 3 mesh running at 200 MHz.

The task graphs have been allocated using the Worst-Fit and Best-Fit bin-packing heuristics[16]. In bin-packing, the objective is to pack a set of items with given sizes into bins. Each bin has a fixed capacity, and items whose total size exceeds this capacity cannot be assigned to the bin. The goal is to minimize the number of bins used. In an analogous way, items may represent tasks, or entire task graphs, while bins represent processors with a given processing capacity. The size of each item corresponds to the

processor utilization by the task. The Best-Fit (BF) strategy places a new object in the bin whose remaining space will be the smallest one. The Worst-Fit (WF) strategy, in turn, places an object in the bin whose remaining space will be the largest one. As a consequence, WF generates a task distribution with load balancing, while BF generates a distribution that is concentrated in some bins.

Ten task graphs were generated with processor utilizations between 5 and 15% each, resulting in an overall utilization of 95% of one core running at 600 MHz speed and 3 V supply voltage. Each task has a WCET of 1 ms in average. We call this the allocation A (all task graphs running in a single processor).

In the following allocations, the task graphs have been distributed over the NoC. In allocations B and C, each task graph has been entirely allocated in a single processor. Allocation B used BF, while allocation C used WF. We call these strategies BF-TG and WF-TG, respectively. A final allocation D also used WF, but considering individual tasks within each graph (and thus using a much finer grain for the allocation).

Table 1. Allocations and utilizations

Core	BF-TG (B)			WF-TG (C)						WF-TK (D)		
	0	1	2	0		1		2		0	1	2
	%	%	%	%	MHz	%	MHz	%	MHz	%	%	%
0	96	93	96	45	90	39	78	36	72	33	33	28
1	-	-	-	33	66	30	60	27	54	33	29	34
2	-	-	-	24	45	20	40	30	60	29	29%	34

Table 1 shows the processor utilizations resulting from these allocations. The table represents the 3x3 matrix of processors in the NoC. With BF-TG (allocation B), only 3 cores were used, with almost 100 % utilization each, while with WF-TG (allocation C) the load has been distributed among all 9 processors. With WF-TK (allocation D), we could achieve a more evenly distributed load than in allocation C, since allocation items (individual tasks) are much smaller. However, allocation D introduces inter-processor communication, since tasks from a single task graph have been distributed over different processors, while allocations B and C have only intra-processor communication.

In the first experiment, allocations A, B, and C are simulated and compared. Allocation A maps all task graphs into a single core running at 600 MHz, while allocation B divides the graphs between 3 cores, which may run now at 200 MHz each. Allocation C has been simulated twice. In the first simulation, a single frequency of 90 MHz is used for all 9 cores. Since Table 1 shows that there is a maximum utilization of 45% for a processor in allocation C, the frequency can be reduced to 45% of the original 200 MHz. In the second simulation of allocation C, the frequencies of the processors

have been individually adjusted from the original 200 MHz, proportionally to the utilizations in Table 1. These individual frequencies are also shown in Table 1 and vary from 40 MHz to 90 MHz.

Table 2 summarizes the results for energy consumption of the first experiment, when task graphs run during 1 second. Results show the impact of frequency operation reduction on system energy consumption. When the frequency is reduced by a factor of 3 (from 600 to 200 MHz), even using 3 cores instead of one (allocation B), the consumption is reduced to 235 mJ (28 % from the energy in allocation A). When the task graphs are divided among 9 cores (first simulation of allocation C), all of them running at 90 MHz, the consumption is reduced to 137 mJ (16% of the value in A). When the frequency of each processor is adjusted to the load it receives, the energy consumption is reduced again to 111 mJ (13% of the value in A). This shows that, when there is a bad load balancing between the processors, it is worthwhile to apply different supply voltages.

Table 2. Experiment 1 results in mJ.

Tasks	(A) on 1 core at 600 MHz	(B) on 3 cores at 200 MHz (A)	(C) on 9 cores at 90 MHz	(C) on 9 cores at different frequencies
Idle	0.00	0.47	4.93	0.44
Scheduling	3.38	0.94	0.27	0.22
Application	842.62	233.59	131.79	110.46
Total	846.00	235.00	137.00	111.12

As expected, allocation C is the best in this first experiment. These results confirm the experiment of Aydin and Yang[17], which shows that a balanced allocation is more energy efficient than a concentrated one, when a voltage scaling approach is applied.

The energy consumption of the scheduler remains with a very low value, due to the relationships between tasks lengths and scheduler length, since the scheduler is called once per task in average. However, this depends on applications and tasks, as well as on the scheduler implementation.

In the second experiment, we compare allocations C (WF-TG) and D (WF-TK), both considering all processors running at 200 MHz. Table 3 shows the average time spent by each processor in different activities (communicating, scheduling, executing the application, or idle). It must be remembered that allocation C concentrates each task graph in a single processor, so that there is no communication between processors. It can be noticed that communication in allocation D consumed 24.45% of the execution time, around 35% of the idle time in allocation C. With an even smaller slack, task deadlines could be missed due to the communication overhead. Scheduling time in allocation D is three times larger than in C, because in this case processors also receive interference from tasks in other processors, such that the scheduler must execute more often.

Table 3. Allocations C and D at 200 MHz.

Task	Allocation C		Allocation D	
	Time (%)	Energy (mJ)	Time (%)	Energy (mJ)
Communication	0.00	0.00	24.45	164.35
Idle	66.88	44.13	36.30	24.65
Scheduling	0.08	0.48	0.26	1.67
Application	33.04	239.38	38.98	289.25
Routers	-	0.00	-	10.09
Total	100.00	283.99	100.00	490.01

The energy consumption increased around 75%, from 284 mJ in allocation C to 490 mJ in D. This is only due to the task allocation strategy and the communication overhead it imposes, since processors have the same frequency. This shows that, considering the inter-processor communication overhead, a good load balancing is not enough for energy minimization.

The energy consumed by the routers is relatively small and does not represent a significant part in this communication overhead.

Even with its communication overhead, when allocation D is compared to the original allocation A (a single processor running all applications and consuming 870 mJ), there is still a reduction of almost 50% in energy consumption. This demonstrates the necessity of mechanisms that combine task allocation strategies that reduce communication overhead with the voltage scaling technique.

6. CONCLUSIONS AND FUTURE WORK

This work introduced a novel model for the simulation of synthetic task graphs that allows the evaluation and comparison of performance and energy consumption of various task allocation strategies in NoC-based systems. This model considers voltage scaling, which must be taken into account by more energy-efficient allocation algorithms, and presents the right abstractions to consider scheduling, communication, and computation costs.

As demonstrated by experiments, two different allocations that have similar timing performances may result in very distinct energy consumptions, because of the communication overhead that may be imposed by the task distribution. It has been also shown that this overhead is mainly imposed to the cores' execution, and not by the NoC itself. However, voltage scaling may considerably mitigate this cost.

Current work includes the development of a static energy consumption model, which will make possible a more precise evaluation of the impact of different task allocation strategies.

This work is a first step in a project aiming at the development and evaluation of on-line task allocation strategies for energy minimization in a

dynamic load environment. The allocation algorithms that have been evaluated are very simple and could be applied on-line, without significant overhead, although they may be still improved. In this context, DVS also needs to be integrated to the scheduler.

REFERENCES

[1] M. Weiser, et al. Scheduling for Reduced CPU Energy. in: Symph. on OS Design and Imp., Monterey CA, 1994, pp. 13-23.

[2] D. Shin, et al. SimDVS: An Integrated Simulation Environment for Performance Evaluation of Dynamic Voltage Scaling Algorithms. in: Workshop on Power-Aware Computer Systems, Springer, Cambridge, MA, 2002, pp. 141-156.

[3] C. Marcon, et al. Modeling the Traffic Effect for the Application Cores Mapping Problem onto NoCs. in: VLSI-SoC, Perth, Australia, 2005, pp.

[4] D. Bertozzi and L. Benini, Xpipes: A Network-on-chip Architecture for Gigascale Systems-on-Chip. IEEE Circuits and Systems Magazine, 4(2): 18-31 (2004).

[5] H. Wang. Orion: A power-performance simulator for interconnection networks. in: ACM MICRO, Istanbul, Turkey, 2002, pp. 294-305.

[6] V. Tiwari, et al., Power analysis of embedded software: a first step towards software power minimization. IEEE Trans. on VLSI Systems, 2(4): 437-445 (1994).

[7] A. C. S. Beck, et al. CACO-PS: A General Purpose Cycle-Accurate Configurable Power Simulator. in: SBCCI, Washington, DC, USA, 2003, pp. 349.

[8] J. Hu and R. Marculescu. Energy-Aware Communication and Task Scheduling for Network-on-Chip Architectures under Real-Time Constraints. in: DATE, Washington, DC, USA, 2004, pp. 10234.

[9] N. S. Kim, et al., Leakage Current: Moore's Law Meets Static Power. Computer, 36(12): 68-75 (2003).

[10] T. T. Ye, et al. Analysis of Power Consumption on Switch Fabrics in Network Routers. in: DAC, New Orleans, 2002, pp. 524-529.

[11] C. A. Zeferino, et al. RASoC: A Router Soft-Core for Networks-on-Chip. in: DATE, 2004, pp. 198-205.

[12] F. Yao, et al. A scheduling model for reduced CPU energy. in: FOCS, Washington, DC, USA, 1995, pp. 374.

[13] R. P. Dick, et al. TGFF: task graphs for free. in: CODES/CASHE, Washington, USA, 1998, pp. 97-101.

[14] M. A. Wehrmeister, et al. Optimizing Real-Time Embedded Systems Development Using a RTSJ-Based API. in: OTM Workshops, 2004, pp. 292-302.

[15] S. A. Ito, et al. Making Java Work for Microcontroller Applications. in: IEEE Design & Test, Los Alamitos, USA, 2001, pp. 100-110.

[16] D. S. Johnson, Near-optimal bin packing algorithms (Cambridge, Mass., 1973).

[17] H. Aydin and Q. Yang. Energy-Aware Partitioning for Multiprocessor Real-Time Systems. in: IPDPS, 2003, pp. 113.